FILTER FOR COMBINED DE-RINGING AND EDGE SHARPENING

BACKGROUND OF THE INVENTION

The present invention is directed to methods and systems for processing digital image signals and more specifically to post-processing methods and systems that reduce quantization noise and coding artifacts in decoded digital images and videos.

An image or video sequence may be converted to electronic signals and stored or transmitted digitally. Encoding an image usually includes estimating approximate visual attributes for small blocks of the image. Encoding also usually includes a data compression ("quantization") scheme for efficiently storing and transmitting a minimum number of bits. The encoding approximations and quantization may degrade the visual quality of the decoded image so a compromise is usually sought between image quality and bit requirements.

The Joint Photographical Expert Group ("JPEG") and the Motion Picture Experts Group ("MPEG") have established standards for the digital representation of images and videos. In the MPEG syntax, for instance, only the bit stream syntax for decoding is specified. This leaves flexibility for the encoder design, which may use standard quantization schemes. An external processor may be used to reduce quantization noise and coding artifacts from a digital signal decompressed by a decoder ("post-processing"). The post-processor is typically a filter that refines and improves displayed images without altering existing quantization schemes, which may be designed into the hardware.

Post-processing systems may now be integrated into new designs for systems that use new quantization standards. Existing quantization schemes, however, may be enhanced by using post-processing filters that can be added to existing systems.

Early digital image compression techniques sought to transmit an image at the lowest possible bit rate and yet reconstruct the image with a minimum loss of perceived quality. These early attempts used information theory to minimize the mean

PDXDOCS:1221505.1

squared error ("MMSE"). But the human visual system ("HVS") does not perceive quality in the MMSE sense, and the classical coding theory of MMSE did not necessarily yield results pleasing to the HVS. Further, classical MMSE theory applied to the human enjoyment of moving video scenes did not yield pleasing results.

For certain wavelengths, the human eye can see a single photon of light in a dark room. This sensitivity of the HVS also applies to quantization noise and coding artifacts within video scenes. The sensitivity of the HVS changes from one part of a video image to another. For example, human sensitivity to quantization noise and coding artifacts is less in the very bright and very dark areas of a video scene (contrast sensitivity). In busy image areas containing high texture or having large contrast or signal variance, the sensitivity of the HVS to distortion decreases. In these busy areas, the quantization noise and coding artifacts get lost in complex patterns. This is known as a masking effect. In smooth parts of an image with low variation, human sensitivity to contrast and distortion increases. For instance, a single fleck of pepper is immediately noticeable and out of place in a container of salt. Likewise, a single contrasting pixel out of place near a strong visual edge in an image may be noticeable and annoying.

The local variance of a video signal is often noticeable to the HVS on a very small scale: from pixel to pixel or from macroblock to macroblock. This means that filters that remove quantization noise and coding artifacts must filter digital data on a scale representing very small blocks of an image. Filters that remove ringing artifacts and sharpen edges of a visual image must perform calculations and operate on each macroblock, or more ideally, on each pixel in a digital image.

In order to convert an image to a digital representation suitable for quantization, an encoder must divide an image into small blocks, each having visual characteristics (such as detail complexity, brightness, and contrast with neighboring blocks) that may be expressed quantitatively for numerical processing. An image is usually partitioned into nonoverlapping blocks having 8 pixels on each side. One scheme for creating a digital representation of an image is discrete cosine transformation ("DCT"). DCT is one method of numerically approximating the visual attributes of an

8x8 pixel block of an image. Partitioning an image into small blocks before applying DCT ("block DCT") reduces computational intensity and memory requirements. This simplifies hardware design. Accordingly, many of the image compression standards available use block DCT coding.

For low bit digital storage and transmission of an image, quantization noise and coding artifacts may appear in the displayed image due to the approximations introduced by block DCT and a quantization scheme, if present. In moving video scenes, these artifacts show as run-time snow and as dirty uncovered backgrounds. Significant artifacts among frames can result in run-time flicker if they are repetitive.

The objectionable artifacts that occur when pictures are coded at low bit rates are color bleeding, blurriness, blockiness, and ringing. Color bleeding is specific to strong chrominance edges. Blurriness is the result of loss of spatial detail in medium-textured and high-textured areas. The two most prevalent coding artifacts are blockiness resulting in loss of edge sharpness, and ringing, the intermittent distortion near visual object boundaries.

Blockiness is the artifact related to the appearance of the 8x8 DCT grid caused by coarse quantization in low-detail areas. This sometimes causes pixelation of straight lines and a loss of edge sharpness. Blockiness occurs when adjacent blocks in an image are processed separately from each other, and coding approximations assigned to each block cause a visual contrast between neighboring blocks that had visual continuity in the original image. For instance, if neighboring blocks lie in an area of the image where intensity is changing, the decoded intensity assigned to each block may not capture the original intensity gradient.

Ringing (also referred to as mosquito noise) occurs at edges on flat backgrounds where high frequencies are poorly quantized. Accordingly, ringing is usually associated with sharp image boundaries, such as text against a uniform background or computer graphics. Coarser quantization block DCT systems are typically ineffective when coding sharp visual edges, so decompressed images usually have

distortion at these edges. Known de-ringing filters blur the true edges when they attempt to remove the ringing artifacts.

The majority of the image (JPEG) and video (MPEG) coding standards are based on the use of block DCT. One serious drawback of compressing images using these standards is the blockiness and ringing artifacts that occur in a decoded image.

Edge sharpening by strengthening the true visual edges in an image is a common post-processing method used to remove artifacts. U.S. Patent No. 5,822,467 to Lopez et al., entitled "Sharpening Filter for Images with Automatic Adaptation to Image Type" is directed to a filtering method that uses Laplacian filter coefficients and normalization divisors. U.S. Patent No. 5,818,972 to Girod et al., entitled "Method and Apparatus For Enhancing Images Using Helper Signals" is directed to a filtering method that uses helper signal architecture and nonlinear characteristic functions. U.S. Patent No. 5,757,977 to Mancuzo et al., entitled "Fuzzy Logic Filter for Reducing Noise and Sharpening Edges of Digital Image Signals" is directed to a filter that reduces noise using a fuzzy logic comparison unit in two noise reduction circuits.

De-ringing is another common post-processing method used to remove artifacts. U.S. Patent No. 5,819,035 to Devaney et al., entitled "Post-filter for Removing Ringing Artifacts of DCT Coding" is directed to a post-filter that performs anisotropic diffusion filtering on decoded data. U.S. Patent No. 5,850,294 to Apostolopoulos et al., entitled "Method and Apparatus for Post-processing Images" is directed to reducing visual artifacts through separate detection, mapping, and smoothing functions. The Apostolopoulos method employs DCT-domain detection rather than edge detection in the pixel domain. U.S. Patent No. 5,883,983 to Yee et al., entitled "Adaptive Post-processing System for Reducing Blocking Effects and Ringing Noise in Decompressed Image Signals" is directed to an adaptive filtering method that uses a binary edge map and filters using various weighting factors to generate the filtered pixel value. One of the drawbacks of known de-ringing filters is that they can result in blurring of the true edges while removing the ringing artifacts.

BRIEF SUMMARY OF THE INVENTION

Although both edge sharpening and de-ringing are known, prior to the present invention they have never been efficiently combined in a single filter that is streamlined and computationally less expensive than using separate edge sharpening and de-ringing filters. The present invention in its preferred embodiments shares data between modules and reuses previous calculations for efficient operation. The present invention seeks to reduce the redundancy involved in separately carrying out edge sharpening and de-ringing. Experimentally, the present invention yields results that are superior to several known de-ringing filters.

The present invention is directed to a filter for post-processing digital images and videos, having an edge mapper, a pixel sorter, and an adaptive filter that simultaneously performs de-ringing and edge sharpening. Combining modules and code simplifies hardware design. Sharing at least some data that has been previously calculated among modules increases speed and efficiency. Thus, in its preferred embodiments, the combined filter is computationally simpler than known methods and can achieve removal of ringing artifacts and sharpening of true edges at the same time.

The edge mapping, de-ringing, and edge sharpening stages of the present invention preferably share data and reuse previous calculations for efficient operation.

The present invention also includes a preferred method of simultaneously de-ringing and edge sharpening digital images and videos.

The foregoing and other objectives, features, and advantages of the invention will be more readily understood upon consideration of the following detailed description of the invention, taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

FIG. 1 is a block diagram of the components of one preferred embodiment of the present invention.

FIG. 2 is a block diagram of preferred components of an edge mapper of the present invention.

FIG. 3 is a block diagram of preferred components of a pixel sorter of the present invention.

FIG. 4 is a block diagram of preferred components of an adaptive filter of the present invention.

FIG. 5 is a flowchart of a preferred method of the present invention.

FIG. 6 is a decompressed black and white image of flowers that has ringing artifacts.

FIG. 7 is a portion of the image of FIG. 6 magnified 220%.

FIG. 8 is the image of FIG. 6 after post-processing with a known de-ringing filter.

FIG. 9 is a portion of the image of FIG. 8 magnified 220%.

FIG. 10 is the image of FIG. 6 after post-processing with a preferred combined de-ringing and edge sharpening filter of the present invention.

FIG. 11 is a portion of the image of FIG. 10 magnified 220%.

DETAILED DESCRIPTION OF THE INVENTION

The present invention presents simple and effective methods and systems for post-processing decoded images by combining separate filtering operations into a single filter. It sharpens true edges and removes distortion from decoded images that typically contain inaccurate visual edges and ringing artifacts. In a post-processing system, the present invention preferably identifies modules common to de-ringing and edge sharpening functions and reuses previously computed data to streamline computational needs and achieve adaptive filtering of an image signal. Reducing redundancy makes this system computationally more efficient than separate de-ringing and edge sharpening filters.

FIG. 1 shows the basic exemplary modules and stages of the present invention in relation to a typical image decoder. A decoder 90, such as a prior art decoder that may use a quantization scheme, decompresses a JPEG, MPEG, or other digital image and sends the decompressed image signal to the present invention 100. An edge mapper 110 receives the signal and creates a binary map of visual edges in the digital image, making the map available to a pixel sorter 120. The pixel sorter 120 preferably routes each pixel signal to one of three filtering possibilities within the adaptive filter 130: the pixel may be assigned to the de-ringing filter 150, to the edge sharpener 160, or to neither. The de-ringing filter will preferably be applied only to those pixels that are not on a true visual edge and are therefore more likely to be ringing artifacts. The edge sharpener will preferably be applied only to pixels that are on a visual edge. In some embodiments, some pixels will receive neither edge sharpening nor de-ringing filtering. Thus, filtering is adaptive.

FIG. 2 shows an exemplary edge mapper 110 of the present invention in detail. An edge detector 200 calculates an edge value for each pixel in a digital image signal received from a decoder 90. The edge detector 200 may use an edge gradient operator 210 including but not limited to at least one Sobel, Prewitt, and/or Roberts edge gradient operator. The shown embodiment preferably uses Roberts operators of the form:

$$H_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad H_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

in order to minimize computational requirements. The use of edge detection operators 210 to compute at least one edge gradient 220 for each pixel in a digital image is well known to a person having ordinary skill in the art.

The edge detector 200 of the present invention preferably computes multiple edge gradients 220 for each pixel. The absolute value of the gradients obtained may be averaged or otherwise weighted into an edge significance value ("EdgeStrength")

by an averager 230. In C++ programming code, EdgeStrength for a given pixel is computed by:

$$\text{EdgeStrength}(i, j) = (|g_{H1}(i, j)| + |g_{H2}(i, j)|)/2;$$

where i and j are coordinates of a pixel within the digital image signal, and $g_{H1}(i, j)$ and $g_{H2}(i, j)$ are gradients returned by the Roberts operators for a pixel at coordinates (i, j).

The edge detector 200 then compares the EdgeStrength of a pixel to a selected threshold value ("EdgeThreshold") using an edge comparator 240. The value of EdgeThreshold, which may be determined experimentally, is selected to optimize the computation of an effective edge map array 250 for use in the later filtering stages. An EdgeThreshold value of 30 is preferred when using Roberts operators for edge detection.

The comparison of a pixel's edge gradient (EdgeStrength) to the selected EdgeThreshold value determines the pixel's edge classification. In one embodiment, if EdgeStrength is less than or equal to EdgeThreshold, then the pixel is classified as a pixel representing a visual non-edge and assigned an edge map value ("EdgeValue") of 0. If EdgeStrength is greater than EdgeThreshold, then the pixel is classified as a pixel representing a visual edge and assigned an edge map value ("EdgeValue") of 1. (The 0 and 1 EdgeValues are meant to be exemplary and may be switched or alternatives used in their place.) The EdgeValue of a pixel may be used immediately by other modules and filtering stages of the present invention, and/or the stored array of edge map values ("EdgeValue(i, j)") may be retained in a discrete step for later use.

In an exemplary C++ programming code embodiment of the invention, the comparison of EdgeStrength to EdgeThreshold and classification of pixels as pixels representing visual edges or pixels representing visual non-edges may be computed by:

```
if (EdgeStrength(i, j) > EdgeThreshold)
{
```

```
EdgeValue(i, j) = 1;
}
else
{
EdgeValue(i, j) = 0;
}.
```

FIG. 3 shows the pixel sorter 120 of the present invention in detail. The sorter preferably ensures that visual edges are not blurred and no true details lost. The pixel sorter 120 preferably accesses the edge map array 250 to retrieve the EdgeValue of the pixel being processed. A first sorting comparator 300 reads the edge map array 250 to determine whether the pixel being processed is a pixel representing a visual edge (EdgeValue=1). If it is, the pixel is routed to the edge sharpener 160.

If the pixel being processed is a pixel representing a visual non-edge (EdgeValue=0) then a selector 310 designates a group of pixels in a grid pattern ("kernel" 320) near each pixel being processed. The purpose of the kernel 320 is to provide the present invention with the visual characteristics of the pixels in the immediate vicinity of the pixel being processed, sometimes called a "region of support." For example, if a pixel being processed stands out from its neighboring pixels, and none of the pixels in the kernel 320 are pixels representing a visual edge, then the likelihood increases that the pixel being processed is manifesting noise, and should be filtered to a tone nearer the tone of its neighboring pixels.

A kernel 320 may be a group of pixels adjacent to, surrounding, or near the pixel being processed. For instance, if the pixel being processed is on the physical edge of the digital image, then the kernel 320 cannot consist of pixels completely surrounding the pixel being processed. The size of the kernel 320 may vary, but is preferably a 3x3 pixel grid with the pixel being processed in the center of the grid. Formulas that produce a kernel grid having a center pixel are used. Exemplary formulas are 2X+1 and 2Y+1 to define each side of the kernel 320. The pixel being processed is

the center pixel in a grid produced by the exemplary formulas. Using 1 for X and Y yields a 3x3 grid; using 2 for X and Y yields a 5x5 grid, etc. A 2x2 pixel grid, for comparison, has no center position and cannot be generated by the 2X+1 and 2Y+1 formulas. Using 1 for X and Y avoids oversmoothing and loss of detail and also keeps the computational complexity low. During operation of the present invention, the kernel may be determined by incrementing the 2X+1 and 2Y+1 formulas in a "for" loop from -X to +X and from -Y to +Y.

The coordinates of the kernel 320 designated by the selector 310 may be used to access the edge map array 250 to determine the EdgeValues of each pixel in the kernel 320. The EdgeValues may be summed 330 to determine the likelihood that the kernel 320 contains a visual edge. In one embodiment, the sum of the EdgeValues from the edge map array 250 is compared by a second comparator 340 to a threshold value that indicates the presence of a true visual edge ("TrueEdgeThreshold"). If the sum of the EdgeValues is less than TrueEdgeThreshold, then it is presumed that the kernel 320 does not contain a visual edge. In one preferred embodiment, a 3x3 pixel grid is used, and TrueEdgeThreshold preferably has a value of 4. This means that if at least four of the nine pixels in the 3x3 grid were mapped by the edge detector 200 as pixels representing a visual edge, then it is assumed that the kernel 320 contains a visual edge. Other variations are possible for determining when a kernel contains a visual edge.

If the kernel 320 containing the pixel being processed contains a visual edge or at least has enough visual edge character to surpass the TrueEdgeThreshold selected value, and the pixel being processed is a visual non-edge pixel, then the pixel sorter 120 preferably classifies the pixel for no filtering 350 and routes the pixel neither to the de-ringing filter 150 nor to the edge sharpener 160. The sorter thereby ensures that the actual edges in the kernel are not blurred and no true details are lost.

If the kernel 320 containing the pixel being processed does not contain a visual edge or at least has so little visual edge character that the TrueEdgeThreshold selected value is not surpassed, then the pixel sorter 120 preferably routes the pixel being processed to the de-ringing filter 150 since the pixel does not lie in a kernel that may

contain a true visual edge. Table 1 summarizes filtration categories to which the pixel sorter 120 may assign individual pixels.

Table 1

Type of Pixel	Type of Filtering Assigned by Pixel Sorter 120
Visual non-edge pixel substantially surrounded by visual non-edge pixels	De-Ringing
Visual non-edge pixel substantially surrounded by visual edge pixels	No Filtering
Visual edge pixel	Edge sharpening

One preferred embodiment of the pixel sorter 120 is the following exemplary C++ programming code routine:

```
SumKernelPixels = 0;

for (ix = -X; ix <= X; ix++)
for (iy = -Y; iy <= Y; iy++)
{
SumKernelPixels += EdgeValue(i + ix, j + jy);
}
if ((SumKernelPixels <  TrueEdgeThreshold) && (EdgeValue(i, j) = 0))
{Apply De-ringing Filter}
else if (EdgeValue(i, j) = 1)
{Apply Edge Sharpener}.
```

In this code, (2X+1)(2Y+1) is the kernel size, and TrueEdgeThreshold is a number less than (2X+1)(2Y+1) (i.e., is a number less than the total number of pixels within the kernel). Only those pixels that satisfy one of the criteria in the shown code will pass through the next stage of actual filtering by the de-ringing filter 150 or the edge sharpener 160.

FIG. 4 shows an exemplary adaptive filter 130 of the present invention in detail. The adaptive filter 130 includes a de-ringing filter 150, an edge sharpener 160, and a no filtering circuit for passing pixel signals that receive no filtering through the adaptive filter 130.

The de-ringing filter 150 receives pixel signals classified for de-ringing by the pixel sorter 120 of the previous stage. Known de-ringing filters adaptively choose a filtering mask based on a pixel's neighboring pixels. A different mask is chosen for each pixel, and the determination of this mask needs several comparison and branching decisions. This can result in slow operation on certain processor architectures. The preferred de-ringing filter of the present invention has lower complexity compared with these known de-ringing filters. Moreover, since the de-ringing filter of the present invention is combined with an edge sharpening filter and supporting modules, the efficiency of the de-ringing filter is enhanced even further by using computation results already performed by the other filters and modules.

The de-ringing filter 150 preferably includes a grayscaler 400 that reads the grayscale values of all pixels in a kernel being processed. The gray scale is a series of gray tones, which range from true black to true white, usually but not always expressed in 255 steps. The grayscaler 400 sums the grayscale values of all pixels representing visual edges in the kernel ("EdgeGrayscaleSum") and sums the grayscale values of all pixels representing visual non-edges in the kernel ("NonEdgeGrayscaleSum").

In an exemplary C++ programming code embodiment of the grayscaler, the computation of the number of pixels representing a visual edge in the kernel

("SumEdgePixels") may be computed at the same time that the grayscale values of the pixels in the kernel are summed:

```
Kernel = (2 * X + 1) * (2 * Y + 1);
NonEdgeGrayscaleSum = EdgeGrayscaleSum = 0;

for(ix = -X; ix <= X; ix++)
for(iy = -Y; iy <= Y; iy++)
{
NonEdgeGrayscaleSum += (1 - EdgeValue(i + ix, j + jy)) * GrayScale(i + ix, j + jy);
EdgeGrayscaleSum += EdgeValue(i + ix, j + jy) * GrayScale(i + ix, j + jy);
SumEdgePixels += EdgeValue(i + ix, j + jy);
}
```

where Kernel is the number of pixels in the kernel containing the pixel being processed, (2X + 1) and (2Y + 1) define the sides of the kernel, GrayScale is the grayscale value of a pixel, NonEdgeGrayscaleSum is the sum of the grayscale values of the pixels representing visual non-edges in the kernel, EdgeGrayscaleSum is the sum of the grayscale values of the pixels representing visual edges in the kernel, EdgeValue is the edge map value (0 or 1) assigned to each pixel by the edge mapper 110, SumEdgePixels is the number of pixels representing visual edges in the kernel, i and j are pixel coordinates within the digital image, and ix and iy are loop incrementing variables. A lookup table may optionally be used to store the values of SumEdgePixels for later use.

The de-ringing filter 150 may also include a weighting module 410 that alters the grayscale value of the pixel being processed in direct proportion to an average grayscale value of all the other pixels representing visual non-edges in the kernel. This weighting of the grayscale of a pixel based on the grayscale characteristics of adjacent pixels representing visual non-edges is one method of performing de-ringing. One

exemplary method of performing the weighting function is by calculating the sum of grayscale values from each pixel representing a visual non-edge in the kernel and dividing the sum by the number of all pixels representing visual non-edges in the kernel.

In an exemplary C++ programming code embodiment of the weighting module, only the summed grayscale values of the pixels representing visual non-edges (NonEdgeGrayscaleSum) are considered, the grayscale values of the pixels representing visual edges do not enter into the computation. The size of the kernel surrounding each pixel being operated on by the de-ringing filter 150 (preferably eight pixels with the pixel being processed being the ninth) does not change during computations, but a unique NonEdgeGrayscaleSum value is computed by the grayscaler 400 for each kernel. The number of pixels in a kernel that represent visual non-edges is used for averaging the sum of the grayscale values, and is given by (Kernel - SumEdgePixels). The grayscaler 400 does not calculate the number of pixels representing visual non-edges by adding EdgeValues, because the EdgeValues of pixels representing visual non-edges are all 0 and cannot be added to give a count. The final grayscale value assigned to a pixel for display may be performed in C++ by:

FinalGrayScale(i, j) = (1/(Kernel - SumEdgePixels)) * NonEdgeGrayscaleSum;

where FinalGrayScale is the grayscale value for final display of each pixel being processed, Kernel is the number of pixels in the kernel, SumEdgePixels is the number of pixels representing a visual edge in the kernel, NonEdgeGrayscaleSum is the sum of the grayscale values of the pixels representing visual non-edges in the kernel, and i and j are pixel coordinates. Although this embodiment performs de-ringing by a simple averaging calculation, other methods of weighting a final grayscale value may be used with the present invention.

The adaptive filter 130 also includes an edge sharpener 160. One preferred method of edge sharpening is unsharp masking. The edge sharpener 160 preferably includes an unsharp masking module 420 that sharpens visual edges by adding

a high pass filtered image of the original digital image to relevant pixels in the original image to yield a final edge sharpened image.

The amount of edge sharpening may be controlled by using an edge sharpening factor ("λ ") for determining the strength of the high pass filtered image to be added to the original image. λ preferably has a value between 0.01 and 0.7 and controls the amount of edge sharpening. A higher λ value can result in edge saturation and added artifacts. A combined clipping and saturation avoidance stage may optionally be added to ensure that there are no added artifacts. Values of $\lambda = 0.5$, 0.25 and 0.125 are also preferable choices since efficient implementation may be achieved by simple shifting.

A high pass filtered image may be obtained by subtracting a low pass filtered image from a scaled original image. This is desirable since a low pass filtered image is readily available from other calculations already performed by the grayscaler 400. For example, dividing the summed grayscale values of the pixels representing visual edges in a kernel (EdgeGrayscaleSum) by the number of pixels representing visual edges in the kernel (SumEdgePixels) gives a low pass filtered image suitable for this embodiment. The low pass filtered image may optionally be multiplied by the edge sharpening factor λ to control the degree of edge sharpening. A computationally efficient method of edge sharpening may be derived algebraically:

Sharpened Image = (Original Image) + (λ)(High Pass Image)

" " = (Original Image) + (λ)(Original Image - Low Pass Image)

" " = (Original Image) + (λ)(Original Image) - (λ)(Low Pass Image)

" " = (1 + λ)(Original Image) - (λ)(Low Pass Image)

The term (1 + λ)(Original Image) provides a scaled version of the original digital image from which a low pass filtered image is subtracted to obtain the sharpened image.

A preferred edge sharpener embodiment using unsharp masking may be performed by the following C++ programming code (together with the code for the de-ringing filter):

FinalGrayScale(i, j) = (1 + λ) * Grayscale(i, j) - (1/SumEdgePixels) * λ * EdgeGrayscaleSum

where GrayScale(i, j) is the grayscale value of the pixel at coordinates i and j before edge sharpening, FinalGrayScale(i, j) is the grayscale value of the edge sharpened pixel at coordinates i and j, λ is the edge sharpening factor to control the strength edge sharpening, SumEdgePixels is the number of pixels representing visual edges in the kernel calculated by the grayscaler 400, and EdgeGrayscaleSum is the sum of the grayscale values of the pixels representing visual edges in the kernel calculated by the grayscaler 400. The FinalGrayScale(i, j) image may optionally be clipped.

The edge sharpener 160 may also include an optional limiter 430 for turning off the edge sharpener 160 if a visual edge is already too strong. One preferred method of determining whether a visual edge is already too strong is to compare EdgeStrength, which is calculated by the edge mapper 110, to a predetermined threshold. If EdgeStrength for a given pixel is greater than the threshold, the edge sharpener will not operate on the pixel. This may avoid saturation of visual edges that are already strong enough.

The various stages and modules of the present invention are discussed as discrete units for ease of explanation, but the different modules are preferably combined and share data in actual embodiments. For example, the code for the pixel sorter 120 and the adaptive filter 130 are combined in preferred embodiments. Combining modules and code simplifies hardware design, and the sharing of data that has been previously calculated by a different module or code fragment increases speed and efficiency.

FIG. 5 shows one preferred method of filtering a digital image. Visual edges in the digital image are mapped to produce an edge map 500. Pixels representing a

visual edge may be distinguished from pixels representing a visual non-edge by using at least one edge gradient operator 510. Pixels representing visual non-edges are preferably sorted for de-ringing 520. A pixel representing a visual non-edge may be sorted for no filtering if the number of pixels representing visual edges in a kernel of pixels surrounding the pixel being processed is greater than a selected edge threshold. If the pixel representing a visual non-edge is sorted for de-ringing, the grayscale value of the pixel is altered in proportion to the averaged grayscale values of pixels in the kernel 540. The de-ringing step preferably uses at least some data previously calculated from the edge mapping and pixel sorting steps. Pixels representing visual edges are preferably sorted for edge sharpening 550. One preferred method of edge sharpening includes unsharp masking by adding a high pass filtered image to the pixel representing a visual edge 560. The edge sharpening step preferably uses at least some data previously calculated from the edge mapping, pixel sorting, and de-ringing steps. Edge saturation may be prevented by limiting edge sharpening 570 of pixels having high edge strength (as calculated by the edge mapper).

Results obtained from the present invention and results obtained from a typical known "de-ringing only" filter are shown in FIGS. 6-11. FIGS. 6-7 show a decompressed image that has ringing artifacts. Blockiness is evident throughout the images. Unsharp edges are also noticeable. FIGS. 8-9 show the images of FIGS. 6-7 after post-processing using a conventional "de-ringing only" filter. There is less blockiness in FIGS. 8-9 than in FIGS. 6-7, but there is a great loss of detail – especially noticeable on the petals of the flower. FIGS. 10-11 show the images of FIGS. 6-7 after post-processing using the combined de-ringing and edge sharpening filter of the present invention. The details on the petals of the flower are more enhanced in FIGS. 10-11 than in FIGS. 8-9, and the ringing artifacts in FIGS. 6-7 are gone. FIGS. 10-11 demonstrate that some of the blocking artifacts evident in FIGS. 6-7 and in FIGS. 8-9 are also removed by the present invention.

It should be noted that although this invention is set forth in terms of a "single filter," conceptually the "single filter" might be divided into a plurality of

interlinked elements such as algorithms, modules, subroutines, or processors. Further, each of the elements may be further divided into a plurality of processors.

It should also be noted that several of the features of the present invention may be replaced with known features. For example, the de-ringing filter 150 may be replaced with a known de-ringing filter or the edge sharpener 160 may be replaced with a known edge sharpener. Although this would make the invention computationally more expensive, it would still fall within the scope of the invention.

The terms and expressions that have been employed in the foregoing specification are used as terms of description, not of limitation, and are not intended to exclude equivalents of the features shown and described or portions of them. The scope of the invention is defined and limited only by the claims that follow.